Attorney Docket No. SUN1P849/P7352

PATENT APPLICATION

ONLINE UPGRADE OF CONTAINER-BASED SOFTWARE COMPONENTS

Inventors:

Masood Seyed Mortazavi
1047 November Drive
Cupertino, CA 95014
Citizenship: USA

Assignee:

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

ONLINE UPGRADE OF CONTAINER-BASED SOFTWARE COMPONENTS

Field of the Invention

[0001] The present invention relates to object oriented computing environments, and more particularly, to techniques for upgrading container-based software components.

Background of the Invention

[0002] Recently, container-based software components have been developed for object oriented computing environments. These software components (e.g., application programs) can interact with a "container" which can typically provide various standard functions (e.g., security, networking, etc.). This offers many advantages, for example, ease of use and reusability.

[0003] To elaborate, in Sun Microsystems' Enterprise JavaBeans component architecture, and in Microsoft Corporation's Component Object Model (COM), a container is an application program or subsystem in which the program building block known as a component is run. For example, a component, such as a button, a small calculator, or a database requestor, can be developed using Enterprise JavaBeans that can run in application servers.

[0004] In today's computing environments, there is often a need to upgrade software components. As such, it is desirable to perform software upgrades in an efficient way. Moreover, for some applications, it is highly desirable to perform software upgrades without having to shut down the system. Unfortunately, conventional techniques do not allow software upgrades to be performed without having to shut down the system or otherwise degrade the performance of the system in some other manner. Typically, services being performed have to be interrupted to allow for the upgrade. In some cases, interruption of services can be very costly, thus need to be avoided.

[0005] In view of the foregoing, there is a need for improved techniques for upgrading software components.

Summary of the Invention

[0006] Broadly speaking, the invention relates to techniques for online upgrading of software components. The invention is especially suited for online upgrading of container-based software components in object oriented computing environments. In accordance with one aspect of the invention, a multi-stage online upgrade system is disclosed. As will be appreciated, the multi-stage online upgrade system can facilitate online installation of the container-based software components (e.g., applications) in object oriented computing environments. Moreover, online software upgrades can be achieved without interrupting online services which are provided by the container-based software components. The multi-stage online upgrade system can be implemented so as to allow interaction with an upgrade management entity (e.g., an application developer or system administrator). This allows controlling and/or monitoring of the online upgrade operations. Other aspects of the invention provide techniques suitable for performing online upgrades of container-based software components. As will be appreciated, online upgrades of the container-based software components can be implemented in multiple stages.

[0007] The invention can be implemented in numerous ways, including as a system, an apparatus, a method, or a computer readable medium. Several embodiments of the invention are discussed below.

[0008] As an object oriented computing environment one embodiment of the invention includes: a first container based software component being an upgraded version of a second container based software component, a container suitable for interaction with the first container based software component, and an online upgrade system capable of operating to facilitate online upgrading of said second container based software component to said first container based software component.

[0009] As a method of upgrading software in a object oriented computing environment, one embodiment of the invention includes the acts of: loading an online upgrade module, notifying an online-upgrade controller to initiate an online upgrade process, and performing one or more operations to facilitate online upgrade of said second container based software component to said first container based software component. The online upgrade module includes a first container based software

component, an online upgrade listener and an online upgrade specification. The first container based software component being an upgrade of a second container based software component.

[0010] As a method of upgrading container based software components in multiple stages one embodiment of the invention includes: an upgrade prepare stage, a pre-upgrade stage, one or more upgrade operations, and a post-upgrade stage.

[0011] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

Brief Description of the Drawings

[0012] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

Fig. 1A illustrates an object oriented computing environment including a multi-stage online upgrade system in accordance with one embodiment of the invention.

Fig. 1B illustrates an exemplary online upgrade package in accordance with one embodiment of the invention.

Fig. 2 illustrates an online upgrade method in accordance with one embodiment of the invention.

Fig. 3 illustrates a method for performing online upgrade operations in accordance with one embodiment of the invention.

Fig. 4 illustrates an exemplary method for performing upgrade prepare (ready) callbacks in accordance with one embodiment of the invention.

Fig. 5 illustrates a method for performing online upgrade operations.

Fig. 6 illustrates a method for performing load and redirect operations upgrade operations.

Fig. 7 illustrates a method for performing commit operations.

Detailed Description Of The Invention

[0013] The invention pertains to techniques for online upgrading of software components. The invention is especially suited for online upgrading of container-based software components in object oriented computing environments. In accordance with one aspect of the invention, a multi-stage online upgrade system is disclosed. As will be appreciated, the multi-stage online upgrade system can facilitate online installation of the container-based software components (e.g., applications) in object oriented computing environments. Moreover, online software upgrades can be achieved without interrupting online services which are provided by the container-based software components. The multi-stage online upgrade system can be implemented so as to allow interaction with an upgrade management entity (e.g., an application developer or system administrator). This allows controlling and/or monitoring of the online upgrade operations. Other aspects of the invention provide techniques suitable for performing online upgrades of container-based software components. As will be appreciated, online upgrades of the container-based software components can be implemented in multiple stages.

[0014] Embodiments of the invention are discussed below with reference to Figs. 1A–7. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes as the invention extends beyond these limited embodiments.

[0015] Fig. 1A illustrates an object oriented computing environment 100 including a multi-stage online upgrade system 102 in accordance with one embodiment of the invention. The multi-stage online upgrade system 102 includes a first container-based software component 104, a multi-stage online upgrade listener interface 106, a multi-stage online upgrade controller 108, a multi-stage online upgrade specification 110, and generated code (artifacts) 112.

[0016] The first container-based software component 104 represents an upgraded version of a second container-based software component 116. The second container-based software component operates in a container 116 of the object oriented computing environment 100. The container 116 can, for example, represent a standard interface which implements various functions (e.g., security, networking, etc.). The first or second container-based software components 104 and

114, for example, can be Enterprise JavaBeans II applications developed in accordance with Sun Microsystems' specifications.

[0017] As will be appreciated, the multi-stage online upgrade system 102 can facilitate online installation of the first container-based software component 104 in the object oriented computing environment 100. Moreover, online software upgrades can be performed without interrupting services that are typically performed by first or second container-based software components 104 and 114. In other words, at least one of the first or second container-based software components 104 and 114 are operable while the multi-stage online upgrade system 102 is upgrading the object oriented computing environment 102 (i.e., replacing the second container-based software component 114 to the first container-based software component 104).

[0018] As noted above, the multi-stage online upgrade system 102 can include the multi-stage online upgrade control module 108. In the described embodiment, the multi-stage online upgrade control module 108 is capable of interacting with the multi-stage online upgrade listener interface 106 to facilitate online installation of the first container-based software component 104. The multi-stage online upgrade listener interface 106 can access the first container-based software component 104, as well as the second container-based software component 114 . It should be noted that at least a portion of the multi-stage online upgrade control module 108 can be implemented in the container 116. Furthermore, the multi-stage online upgrade control module 108 can be in communication with a program developer or system administrator 118. In this way, online upgrading can be controlled and/or monitored by a human operator.

[0019] It should also be noted that the multi-stage online upgrade specification 110 can provide information regarding the upgrade. The generated code (artifacts) 112 can represent the code which is generated for the operation of the second container-based software component 104. As will be appreciated, when the online upgrade operations have successfully been completed and the first container-based software component 104 is fully operable the second container-based software component 114 can be removed. However, it should be noted that first and second container-based software components 104 and 114 may simultaneously be operable during the online upgrading process. As will be appreciated, this means that it is

possible to "roll back" to the second container-based software component 114 during the online upgrading process, for example, in the event the upgrade is unsuccessful or the upgrade process is terminated. In any case, the multi-stage online upgrade system 102 allows software components to be upgraded without interruption to services provided by them.

[0020] As will be appreciated, one or more components of the multi-stage online upgrade system 102 can be packed in a software package. Fig. 1B illustrates an exemplary online upgrade package 150 in accordance with one embodiment of the invention. The online upgrade package 150 includes an Enterprise JavaBeans II application 152, an upgrade listener interface 154, and a manifest 156 which includes an upgrade specification 158. It should be noted that online upgrade package 150 can also include a generated code (artifact) 160.

[0021] Fig. 2 illustrates an online upgrade method 200 in accordance with one embodiment of the invention. Initially, at operation 202, an online upgrade package is loaded. The online upgrade package includes a container-based software component. Typically, the container-based software component needs to be installed to upgrade a computing environment with a newer version of a software component. As noted above, an online upgrade package can also include other components, for example, an upgrade listener interface and a manifest which includes an upgrade specification.

[0022] After the online upgrade package is loaded, an upgrade controller (e.g., multi-stage online upgrade control module 108 of Fig. 1A) is notified at operation 204 so that the online upgrading of a software component can be initiated. Next, at operation 206, one or more online upgrade operations are performed to upgrade the computing environment with the container-based software component of the online upgrade package. The online upgrade method 200 ends following operation 206.

[0023] As noted above, the online upgrade operations can be performed in two or more stages. Fig. 3 illustrates a method 300 for performing online upgrade operations in accordance with one embodiment of the invention. The method 300 represents on line operations that can be performed, for example, by the operation 206 of Fig. 2. Initially, at operation 302, one or more upgrade prepare (ready) operations are performed. Next, at operation 304, one or more pre-upgrade

operations are performed. Thereafter, at operation 306, one or more upgrade operations are performed. After the upgrade operations, one or more post-upgrade operations are performed at operation 308.

[0024] Next, at operation 310, a determination is made as to whether the software upgrade is ready for service. If it is determined at operation 310 that the software upgrade is ready for service, the method 300 proceeds to operation 312 where one or more commit operations are performed. As will be appreciated, after the commit operations are performed, the old version of the software can be taken offline and/or removed. The method 300 ends following operation 312.

[0025] On the other hand, if it is determined at operation 310 that the upgrade software is not ready for service, the method 300 proceeds to operation 314 where a determination is made as to whether any rollback operations should be performed. If it is determined at operation 314 that there is no need to perform any rollback operations, the method 300 proceeds to operation 310 where it is determined whether the software is ready for service. However, if it is determined at operation 314 that at least one rollback operation should be performed, the method 300 proceeds to operation 316 where one or more rollback operations are performed. As will be appreciated by those skilled in the art, the rollback operations, among other things, can ensure the integrity of the computing environment (e.g., integrity of a data). After operation 316, the method 300 proceeds to operation 318 where a determination is made as to whether to terminate the online upgrade process. The method 300 ends if it is determined at operation 318 that the online upgrade process should be terminated. However, if it is determined at operation 318 that the online upgrade process should not be terminated, the method 300 proceeds to operation 310 where it is determined whether the software upgrade is ready for service. Thereafter, the method 300 proceeds in the same manner as described above. The method 300 ends following the commit operations 312 or after operation 318 if it is determined that online upgrade processing operations should be terminated.

[0026] Fig. 4 illustrates an exemplary method 400 for performing upgrade prepare (ready) callbacks in accordance with one embodiment of the invention. The method 400 represents, for example, the operations that can be performed by the upgrade prepare stage 302 of Fig. 3. In the described embodiment, the method 400 demonstrates interactions between a server (e.g., an Enterprise JavaBeans II

server) and an online upgrade listener (e.g., multi-stage online upgrade listener interface 106 of Fig. 1A). As will be appreciated by those skilled in the art, the server can be implemented, for example, as a cluster (i.e., two or more computing nodes that can be coupled together).

[0027] At operation 402, online upgrade listener classes are loaded and a listener is instantiated. Next, at operation 404, a determination is made as to whether the online upgrade listener classes are successfully loaded and the listener is successfully instantiated. If it is determined at operation 404 that the online upgrade listener classes are not successfully loaded or the listener is not successfully instantiated, the method 400 proceeds to operation 406 where the results are conveyed to the cluster management, allowing the cluster management to take appropriate action.

[0028] However, if the online upgrade listener classes are successfully loaded and the listener is successfully instantiated, the method 400 proceeds to operation 408 where upgrade prepare callbacks are performed. In addition, at operation 409, the old version of the application is alerted that the online upgrading is about to begin. Next, at operation 410, a determination is made as to whether the upgrade prepare callbacks were successfully performed. If it is determined at operation 410 that the callbacks were successfully performed, the method 400 proceeds to operation to 412 where the upgrade sequence is initiated. However, if it is determined at operation 410 that the callbacks were not successfully performed, the method 400 proceeds to operation 406 where the results are conveyed to the cluster management, this allowing the cluster management to take appropriate action (e.g., terminate the upgrade prepare stage, restart the upgrade ready stage, etc.).

[0029] Fig. 5 illustrates a method 500 for performing online upgrade operations. In the described embodiment, the upgrade operations include: pre-upgrade callbacks, upgrade operations, and post-upgrade operations. The method 500 represents, for example, the operations that can respectively be performed at the pre-upgrade, upgrade, and post-upgrade stages 304, 306 and 308 of Fig. 3. In the described embodiment, the method 500 demonstrates interactions between a server (e.g., an Enterprise JavaBeans II server) and an online upgrade listener (e.g., the multi-stage online upgrade listener 106 of Fig. 1A). Again, the server can be

implemented, for example, as a cluster (i.e., two or more computing nodes that can be coupled together).

[0030] Initially, at operation 502, one or more pre-upgrade callbacks are performed. In addition, at operation 504, the signal prior to upgrade operation is handled. Next, at operation 506, a determination is made as to whether the pre-upgrade callbacks have been performed successfully. If it is determined at operation 506 that the pre-upgrade callbacks were not performed successfully, the method 500 proceeds to operation 508 where the results are conveyed to the cluster management, allowing the cluster management to take appropriate action. However, if it is determined at operation 506 that the pre-upgrade callbacks have been performed successfully, the method 500 proceeds to operation 510 where one or more upgrade operations are performed. These upgrade operations, for example, can be schema expansion or schema contraction for a database application.

[0031] Next, at operation 512, a determination is made as to whether the upgrade operations were performed successfully. If it is determined at operation 512 that the upgrade operations were not performed successfully, the method 500 proceeds to operation 508 where the results are conveyed to the cluster management. However, if it is determined at operation 512 that the upgrade operations were performed successfully, the method 500 proceeds to operation 514 where one or more post-upgrade call backs. In addition, at operation 515, the signal posterior to the upgrade operation is handled.

[0032] Thereafter, at operation 516, a determination is made as to whether the post-upgrade callbacks were performed successfully. If it is determined at operation 516 that the post-upgrade callbacks were not performed successfully, the method 500 proceeds to operation 508 where the results are conveyed to the cluster management. However, if it is determined at operation 516 that the post-upgrade callbacks were performed successfully, the method 500 proceeds to operation 518 where loading of the new version of the application is initiated.

[0033] Fig. 6 illustrates a method 600 for performing load and redirect operations upgrade operations. The method 600 represents, for example, the operations that can be performed at operation 310 Fig. 3. In the described embodiment, the method 600 demonstrates interactions between a server (e.g., an

Enterprise JavaBeans II server) and an online upgrade listener (e.g., multi-stage online upgrade listener interface 106). Again, the server can be implemented, for example, as a cluster.

[0034] At operation 602, the new version of the application is loaded. Next, at operation 604, a determination is made as to whether the new version of the application was loaded successfully. If it is determined at operation 604 that the new version of the application was not loaded successfully, the method 600 proceeds to operation 606 where the results are conveyed to the cluster management. However, if it is determined at operation 606 that the new version of the application was loaded successfully, the method 600 proceeds to operation 608 where a "is-ready-for-service" callback is performed. In addition, at operation 609, the "is-ready-for-service" signal is handled.

[0035] Next, at operation 610, a determination is made as to whether the "is-ready-for-service" callback was performed successfully. If it is determined at operation 610 that the "is-ready-for-service" callback was not performed successfully, the method 600 proceeds to operation 606 where the results are conveyed to the cluster management. However, if it is determined at operation 610 that the "is-ready-for-service" callback was performed successfully, the method 600 proceeds to operation 612 where a redirect callback is performed. Next, at operation 614, it is determined whether the redirect callback is performed successfully. Thereafter, at operation 606, the result of the determination made at operation 614 are conveyed to the cluster management.

[0036] Fig. 7 illustrates a method 700 for performing commit operations. The method 700 represents, for example, the operations that can be performed at operation 312 of Fig. 3. In the described embodiment, the method 700 demonstrates interactions between a server (e.g., a Enterprise s II server) and an online upgrade listener (e.g., multi-stage online upgrade listener interface 106 of Fig. 1A, or upgrade listener interface 154 of Fig. 4). The server can be implemented, for example, as a cluster. Initially, at operation 702, it is assured that the old version of the application is drained. Next, at operation 704, a determination is made as to whether the drainage of the old version of the application was successful. If it is determined at operation 704 that the drainage of the old version of the application was not successful, the method 700 proceeds to operation 706 where the results are

conveyed to the cluster management. However, if it is determined at operation 704 that the drainage of the old version of the application was successful, the method 700 proceeds to operation 708 where the upgrade commit callback is performed. In addition, at operation 709, the upgrade commit signal is handled.

[0037] Next, at operation 710, a determination is made as to whether the upgrade commit callback has been performed successfully. If it is determined at operation 710 that the upgrade commit callback has been not performed successfully, the method 700 proceeds to operation 706 where the results are conveyed to the cluster management. However, if it is determined at operation 710 that the upgrade commit callback has been performed successfully, the method 700 proceeds to operation 712 where the old version of the application is unloaded. Thereafter, at operation 714, a determination is made as to whether the old version of the application was unloaded successfully. Accordingly, at operation 714, the results are conveyed to the cluster management.

[0038] The invention can use a combination of hardware and software components. The software can be embodied as computer readable code (or computer program code) on a computer readable medium. The computer readable medium is any data storage device that can store data which can thereafter be read by a computer system. Examples of the computer readable medium include read-only memory, random-access memory, CD-ROMs, magnetic tape, and optical data storage devices. The computer readable medium can also be distributed over network-coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

What is claimed is: